

Order-Sorted Dependency Pairs*

Salvador Lucas

DSIC, Universidad Politécnica de Valencia, Spain
slucas@dsic.upv.es

José Meseguer

CS dept., University of Illinois at Urbana-Champaign,
USA
meseguer@cs.uiuc.edu

Abstract

Types (or *sorts*) are pervasive in computer science and in rewriting-based programming languages, which often support *subtypes* (subsorts) and subtype polymorphism. Programs in these languages can be modeled as *order-sorted* term rewriting systems (OS-TRSs). Often, termination of such programs heavily depends on sort information. But few techniques are currently available for proving termination of OS-TRSs; and they often fail for interesting OS-TRSs. In this paper we generalize the dependency pairs approach to prove termination of OS-TRSs. Preliminary experiments suggest that this technique can succeed where existing ones fail, yielding easier and simpler termination proofs.

Categories and Subject Descriptors D.1.1 [PROGRAMMING TECHNIQUES]: Applicative (Functional) Programming; D.2.4 [SOFTWARE ENGINEERING]: Software/Program Verification; D.3.2 [PROGRAMMING LANGUAGES]: Language Classifications; D.3.3 [PROGRAMMING LANGUAGES]: Language Constructs and Features; F.3.1 [LOGICS AND MEANINGS OF PROGRAMS]: Specifying and Verifying and Reasoning about Programs; F.3.2 [LOGICS AND MEANINGS OF PROGRAMS]: Semantics of Programming Languages; F.3.3 [LOGICS AND MEANINGS OF PROGRAMS]: Studies of Program Constructs

General Terms Theory, Verification

Keywords Program analysis, term rewriting, termination.

1. Introduction

Types (or *sorts*) are pervasive in computer science and in programming languages. In particular in rewriting-based programming languages like, for example, CAFE OBJ (14), ELAN (6), HASKELL (28), MAUDE (7) and OBJ (24) programmers usually write programs where functions take arguments and return values of particular types. Also, such programs often have *subtypes* (subsorts), and use *subtype polymorphism*, allowing the same symbol, e.g., $+$, to apply to several subtype-related types like *Nat*, *Int*, and *Rat*. Therefore, these programs are naturally modeled as *many-sorted* or, more

* Salvador Lucas was partially supported by the EU (FEDER) and the Spanish MEC, under grant TIN 2007-68093-C02-02; José Meseguer was partially supported by ONR Grant N00014-02-1-0715.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PPDP'08, July 15–17, 2008, Valencia, Spain.

Copyright © 2008 ACM 978-1-60558-117-0/08/07...\$5.00

generally, *order-sorted* TRSs. Here many-sorted TRSs (MS-TRSs) are TRSs enriched with a set S of *sorts* which is used to give the function symbols f a rank $s_1 \cdots s_k \rightarrow s$ which limits the (many-sorted) terms which can be used as arguments of f and to indicate which is the sort of a term t whose root symbol is f . Order-sorted TRSs (OS-TRSs) are MS-TRSs whose set of sorts S comes with a *subsort ordering*, which is interpreted in the models as set inclusion. Furthermore, the function symbols can be *subsort overloaded*, like $+: Nat\ Nat \rightarrow Nat$, $+: Int\ Int \rightarrow Int$, where $Nat < Int$, supporting so-called subtype polymorphism.

Rewriting techniques for MS- and OS-TRSs are well-known (see (22; 23), for instance). But termination of MS- or OS-TRSs remains for the most part underexplored. Only few works address the problem of proving termination of OS-TRSs (21; 40; 12), whereas termination of MS-TRSs has been mostly investigated as a termination technique for TRSs (43) (but see also (4; 13; 32)). However, termination can crucially depend on the sort information.

EXAMPLE 1. Consider the OS-TRSs¹ in Figure 1. As proved by Gnaedig, viewed as an OS-TRS, **EVEN** is terminating (21; 40). However, if *sorts* are removed, the resulting TRS is nonterminating (due to the last rule for **is-even**). The same happens with **FACTORIAL** (due to the last **fact** rule); furthermore, it cannot be proved terminating using existing methods like (21; 40), but it can be proved terminating with our OS-DP method (see Example 15 below).

To prove an OS-TRS terminating, two main approaches can be taken: one can either use *transformational* methods, such as those in (40; 12; 35), to transform an OS-TRS into a TRS, or one can try to develop *intrinsic* methods, which keep the sort information of the OS-TRS, and do not require any transformation. So far, almost all methods for proving termination of programs written in declarative programming languages like CAFE OBJ, ELAN, HASKELL, MAUDE/OBJ, or PROLOG (38) translate them into (variants of) term rewriting systems (TRSs (42)), and then use transformational techniques and tools for proving termination of rewriting: see (12; 17; 34; 41) for recent proposals applicable to some of the aforementioned programming languages. In comparison, intrinsic methods are considerably less developed. As for OS-TRSs, the only two intrinsic methods we are aware of in this context are the extension of the Lexicographic Path Ordering (LPO (29)) by Gnaedig (21), and, in the many-sorted subcase, the technique of many-sorted interpretations proposed in (13). Intrinsic methods work generally better for two main reasons: (i) they *do not lose any information*, and (ii) they tend to lead to *simpler* proofs. Our proposed method is intrinsic: it generalizes the *dependency pairs method* (5) so that it can be used directly to reason about the termination of OS-TRS in an intrinsic way. In particular, our method can prove termination of

¹ Since MAUDE provides an appropriate (self-explanatory) notation for OS-TRSs, and we actually need to give all the details of each specification, the OS-TRSs that we discuss in this paper are presented as MAUDE modules.

```

fmod EVEN is
  sorts Zero NzNeg Neg NzPos Pos Int Bool .
  subsorts Zero < Neg < Int .
  subsorts Zero < Pos < Int .
  subsorts NzNeg < Neg .
  subsorts NzPos < Pos .
  op 0 : -> Zero .
  op s : Pos -> NzPos .
  op p : Neg -> NzNeg .
  op true : -> Bool .
  op false : -> Bool .
  op is-even : Int -> Bool .
  op is-even : NzPos -> Bool .
  op is-even : NzNeg -> Bool .
  op opposite : NzNeg -> NzPos .
  var X : Pos .
  var Y : NzNeg .
  eq is-even(0) = true .
  eq is-even(s(0)) = false .
  eq is-even(s(s(X))) = is-even(X) .
  eq is-even(Y) = is-even(opposite(Y)) .
  eq opposite(p(0)) = s(0) .
  eq opposite(p(Y)) = s(opposite(Y)) .
endfm

```

```

fmod FACTORIAL is
  sorts Nat NzNat .
  subsorts NzNat < Nat .
  op 0 : -> Nat .
  op s : Nat -> NzNat .
  op p : NzNat -> Nat .
  op +_ : Nat Nat -> Nat .
  op +_ : NzNat Nat -> NzNat .
  op +_ : NzNat NzNat -> NzNat .
  op *_ : Nat Nat -> Nat .
  op *_ : NzNat NzNat -> NzNat .
  op fact : Nat -> NzNat .
  vars x y : Nat .
  vars x' : NzNat .
  eq x + 0 = x .
  eq x + s(y) = s(x + y) .
  eq x * 0 = 0 .
  eq x * s(y) = x + (x * y) .
  eq fact(0) = s(0) .
  eq fact(x') = x' * fact(p(x')) .
  eq p(s(x)) = x .
endfm

```

Figure 1. Two order-sorted TRSs in Maude syntax

*subtype polymorphic*² functions in an intrinsic way. For example, the function `is-even` in the `EVEN` module, and the addition and multiplication functions in the `FACTORIAL` module are all subtype polymorphic.

Given a TRS \mathcal{R} , the dependency pairs of \mathcal{R} give rise to a new TRS $\text{DP}(\mathcal{R})$ which (together with \mathcal{R}) determines the so-called *dependency chains* whose finiteness or infiniteness characterize termination or non-termination of \mathcal{R} . Given a rewrite rule $l \rightarrow r$, we get dependency pairs $l^\sharp \rightarrow s^\sharp$ for all subterms s of r which are rooted by a defined symbol; the notation t^\sharp for a given term t means that the root symbol f of t is *marked*, thus becoming f^\sharp (often written F). A chain of dependency pairs is a sequence $u_i \rightarrow v_i$ of dependency pairs such that there is a substitution σ satisfying that $\sigma(v_i)$ rewrites to $\sigma(u_{i+1})$ for all $i \geq 1$. Dependency pairs have an associated *dependency graph*, where absence of infinite chains can be analyzed by considering the (maximal) *cycles* in the graph (5; 20). All these basic intuitions generalize to OS-TRSs, with some important differences. For instance, the dependency pair

$$\text{IS-EVEN}(Y) \rightarrow \text{IS-EVEN}(\text{opposite}(Y))$$

(with `IS-EVEN` a new symbol taking arguments of sort `NzNeg`) belongs to the set of *order-sorted dependency pairs* that we define below. However, in sharp contrast with the unsorted case, it does *not* generate the *infinite chain*

$$\begin{aligned} \text{IS-EVEN}(Y) &\rightarrow \text{IS-EVEN}(\text{opposite}(Y)) \\ &\rightarrow \text{IS-EVEN}(\text{opposite}(\text{opposite}(Y))) \\ &\rightarrow \dots \end{aligned}$$

`opposite(Y)` has sort `NzPos`, and `Y` sort `NzNeg`, which are *incomparable* in the subsort ordering: no *order-sorted rewriting step* is possible beyond the first. Since a TRS is a 1-sorted MS-TRS, and a MS-TRS is an OS-TRS with discrete poset of sorts, our method includes the standard DP method, and a MS-DP method for proving termination of MS-TRSs as special cases.

Section 2 contains preliminaries, Section 3 introduces order-sorted dependency pairs. Section 4 explains the use of order-sorted dependency pairs (OS-DPs) for proving termination of OS-TRSs. We prove correctness and completeness: OS-DPs fully characterize termination of (sort-decreasing) OS-TRSs. In Section 5 we define order-sorted dependency graphs (OS-DGs) and show how to approximate the (usually noncomputable) OS-DG. In Sections 6, 7, and 8 we show how to prove termination of OS-TRSs by means of the new approach, including the use of reduction pairs, the narrowing transformation, and the subterm criterion. Section 9 provides a first experimental evaluation of our techniques. Section 10 covers related work and conclusions.

2. Preliminaries

We summarize here material from (23; 30) on order-sorted algebra and order-sorted rewriting. We start with a partially ordered set (S, \leq) of *sorts*, where $s \leq s'$ is interpreted as *subsort inclusion*. The *connected components* of (S, \leq) are the equivalence classes $[s]$ corresponding to the least equivalence relation \equiv_{\leq} containing \leq . We also define $\lfloor s \rfloor = \{s' \in S \mid s' \leq s\}$, i.e., the sorts in S which are smaller than or equal to s . When $\lfloor s \rfloor$ has an upper bound, we denote it by $\top_{\lfloor s \rfloor}$. An order-sorted signature (Σ, S, \leq) consists of a poset of sorts (S, \leq) and a $S^* \times S$ -indexed family of sets $\Sigma = \{\Sigma_{w,s}\}_{(w,s) \in S^* \times S}$, which are *function symbols* with a given string of argument sorts and a result sort. If $f \in \Sigma_{s_1 \dots s_n, s}$, then we display the function symbol f as $f : s_1 \dots s_n \rightarrow s$. This is called a *rank declaration* for symbol f . Some of these symbols f can be *subsort-overloaded*, i.e., they can have several rank declarations related in the \leq ordering (23). Constant symbols, however, have only one rank declaration.

Given an S -sorted set $\mathcal{X} = \{\mathcal{X}_s \mid s \in S\}$ of *disjoint* sets of variables, the set $\mathcal{T}(\Sigma, \mathcal{X})_s$ of terms of sort s is the least set such that $\mathcal{X}_s \subseteq \mathcal{T}(\Sigma, \mathcal{X})_s$; if $s' \leq s$, then $\mathcal{T}(\Sigma, \mathcal{X})_{s'} \subseteq \mathcal{T}(\Sigma, \mathcal{X})_s$; and if $f : s_1 \dots s_n \rightarrow s$ is a rank declaration for symbol f and $t_i \in \mathcal{T}(\Sigma, \mathcal{X})_{s_i}$ for $1 \leq i \leq n$, then $f(t_1, \dots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})_s$. The set $\mathcal{T}(\Sigma, \mathcal{X})$ of order-sorted terms is $\mathcal{T}(\Sigma, \mathcal{X}) =$

² See Section 10 for a brief discussion of *parametric polymorphism*.

$\cup_{s \in S} \mathcal{T}(\Sigma, \mathcal{X})_s$. An element of any set $\mathcal{T}(\Sigma, \mathcal{X})_s$ is called a *well-formed* term. A simple syntactic condition on (Σ, S, \leq) called *preregularity* (23) ensures that each well-formed term t has always a *least-sort* possible among all sorts in S , which is denoted $LS(t)$. Terms are viewed as labelled trees in the usual way. Positions p, q, \dots are represented by chains of positive natural numbers used to address subterm positions of t . The set of positions of a term t is denoted $\mathcal{Pos}(t)$. Positions of non-variable symbols in t are denoted as $\mathcal{Pos}_\Sigma(t)$, and $\mathcal{Pos}_\mathcal{X}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$, and $t[u]_p$ is the term t with the subterm at position p replaced by u . We write $t \succeq u$, read u is a *subterm of* t , if $u = t|_p$ for some $p \in \mathcal{Pos}(t)$ and $t \triangleright u$ if $t \succeq u$ and $t \neq u$.

An order-sorted substitution σ is an S -sorted mapping $\sigma = \{\sigma : \mathcal{X}_s \rightarrow \mathcal{T}(\Sigma, \mathcal{X})_s\}_{s \in S}$ from variables to terms. The application of an OS-substitution σ to t (denoted $\sigma(t)$) consists of simultaneously replacing the variables occurring in t by a term according to the mapping σ . A specialization ν is an injective OS-substitution that maps a variable x of sort s to a variable x' of sort $s' \leq s$.

An (order-sorted) rewrite rule is an ordered pair (l, r) , written $l \rightarrow r$, with $l, r \in \mathcal{T}(\Sigma, \mathcal{X})$, $l \notin \mathcal{X}_s$ for all $s \in S$, $\text{Var}(r) \subseteq \text{Var}(l)$ (and $LS(l) \equiv_{\leq} LS(r)$ for order-sorted rules). If for all specializations ν , $LS(\nu(l)) \geq LS(\nu(r))$, then we say that the OS-rule $l \rightarrow r$ is *sort-decreasing*. An OS-TRS is a pair $\mathcal{R} = (\Sigma, R)$ where R is a set of OS-rules. We say that \mathcal{R} is sort-decreasing if all rules in R are so. A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ rewrites to u (at position $p \in \mathcal{Pos}(t)$ and using the rule $l \rightarrow r$), written $t \xrightarrow{p, l \rightarrow r} u$ (or just $t \rightarrow_{\mathcal{R}} u$ or even $t \rightarrow u$ if no confusion arises), if $t|_p = \sigma(l)$ and $u = t[\sigma(r)]_p$, for some OS-substitution σ ; if \mathcal{R} is *not* sort-decreasing, we also require the existence of a sort s and a variable $x \in \mathcal{X}_s$ such that $t[x]_p$ is well-formed and $\sigma(r) \in \mathcal{T}(\Sigma, \mathcal{X})_s$. According to this, we have the following.

PROPOSITION 1. *Let \mathcal{R} be a sort-decreasing OS-TRS, $t, u, v \in \mathcal{T}(\Sigma, \mathcal{X})$ and $p \in \mathcal{Pos}(t)$. If $t = t[u]_p$ and $u \rightarrow v$, then $t[u]_p \rightarrow t[v]_p$.*

Without sort-decreasingness, this important result does not hold.

EXAMPLE 2. *Consider the following nonterminating OS-TRS (a variant of (30, Example 1)):*

```
fmod EXAMPLE_2 is
  sorts S S' .
  subsorts S' < S .
  op a : -> S' .
  op b : -> S .
  op f : S -> S .
  op f : S' -> S' .
  op h : S' -> S' .
  var X : S .
  eq f(X) = f(b) .
endfm
```

which is not sort-decreasing (specializing variable X to \mathbf{a} in the rule leads to $LS(\mathbf{f}(\mathbf{a})) = \mathbf{S}' \not\leq \mathbf{S} = LS(\mathbf{f}(\mathbf{b}))$). We have that $\mathbf{f}(\mathbf{a}) \rightarrow \mathbf{f}(\mathbf{b})$, but $\mathbf{h}(\mathbf{f}(\mathbf{a}))$ is a normal form because no reduction on $\mathbf{f}(\mathbf{a})$ is possible within context $\mathbf{h}(\square)$.

Terms $t, t' \in \mathcal{T}(\Sigma, \mathcal{X})$ unify if there is an order-sorted substitution θ such that $\theta(t) = \theta(t')$. A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ *narrowes* to a term t' (written $t \xrightarrow{p} t'$), if there exists a non-variable position $p \in \mathcal{Pos}_\Sigma(t)$ and a rule $l \rightarrow r$ in \mathcal{R} (sharing no variable with t) such that $t|_p$ and l unify with an order-sorted unifier θ , and $t' = \theta(t[r]_p)$ is a well-formed term.

3. Dependency Pairs for proving termination of OS-TRS

Given a TRS \mathcal{R} , the *minimal* nonterminating terms (associated to \mathcal{R}) are nonterminating terms t whose proper subterms u (i.e., $t \triangleright u$) are terminating. \mathcal{T}_∞ is the set of minimal nonterminating terms associated to (the intended TRS) \mathcal{R} . Minimal nonterminating terms are always rooted by a *defined* symbol $f \in \mathcal{D}$; as usual, we say that a symbol $f \in \mathcal{F}$ is *defined* (with respect to a TRS \mathcal{R}) if it occurs at the topmost position of the left-hand side of a rewrite rule in \mathcal{R} , i.e., there is a rule $l \rightarrow r$ in \mathcal{R} with $l = f(l_1, \dots, l_k)$ for some terms l_1, \dots, l_k . Symbols which are not defined in this sense are often called *constructor symbols*; let $\mathcal{C} = \mathcal{F} - \mathcal{D}$ be the set of constructor symbols of a TRS \mathcal{R} . Note that every nonterminating term t contains a minimal nonterminating term u such that $t \succeq u$, see, e.g., (25; 26).

The notion of dependency pair arises by considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term $t = f(t_1, \dots, t_k)$. Such sequences proceed as follows (see, e.g., (25)):

1. a finite number of reductions can be performed *below* the root of t , thus rewriting t into t' ; then
2. a rule $f(l_1, \dots, l_k) \rightarrow r$ applies *at the root* of t' (i.e., $t' = \sigma(f(l_1, \dots, l_k))$ for some substitution σ); and
3. there is a minimal nonterminating term $u = g(s_1, \dots, s_m)$ at some position p of $\sigma(r)$ satisfying that $p \in \mathcal{Pos}_{\mathcal{F}}(r)$, i.e., p is a *nonvariable position of* r .

This means that considering the occurrences of defined symbols in the right-hand sides of the rewrite rules suffices to ‘catch’ *every possible infinite rewrite sequence starting from* $\sigma(r)$. In particular, no infinite sequence can be issued *below* the variables of r (more precisely: all bindings $\sigma(x)$ are *terminating* terms). This is summarized as follows (where $\xrightarrow{\Delta}$ means non-top-reduction, i.e., all rewriting steps are performed *below* the root of the term, and $\xrightarrow{\Delta}$ means top-reduction, i.e., all rewriting steps are performed at the root of the term).

PROPOSITION 2. (25, Lemma 1) *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS. For all $t \in \mathcal{T}_\infty$, there exist $l \rightarrow r \in R$, a substitution σ and a term $u \in \mathcal{T}_\infty$ such that $\text{root}(u) \in \mathcal{D}$, $t \xrightarrow{\Delta^*} \sigma(l) \xrightarrow{\Delta} \sigma(r) \succeq u$ and there is a non-variable subterm v of r , $r \succeq v$, such that $u = \sigma(v)$.*

In the following, $\mathcal{T}_{\Sigma, \infty}$ is the set of minimal nonterminating OS-terms associated to (the intended OS-TRS) \mathcal{R} , i.e., nonterminating OS-terms t whose proper subterms u (i.e., $t \triangleright u$) are terminating.

REMARK 1. *Note that, in sharp contrast to the unsorted case, terminating terms t can contain nonterminating subterms u . For instance, consider the OS-TRS \mathcal{R} in Example 2. Then, $\mathbf{h}(\mathbf{f}(\mathbf{a}))$ is a normal form but $\mathbf{f}(\mathbf{a})$ is nonterminating.*

Using Proposition 1, we can prove the following.

PROPOSITION 3. *Let \mathcal{R} be a sort-decreasing OS-TRS and $t \in \mathcal{T}(\Sigma, \mathcal{X})$ be a terminating term. Then, for all $u \in \mathcal{T}(\Sigma, \mathcal{X})$ such that $t \triangleright u$, u is terminating.*

The *underlying* TRS of an OS-TRS $\mathcal{R} = (\Sigma, R)$ is the TRS $\Theta(\mathcal{R}) = (\mathcal{F}, \Theta(R))$ which is obtained from \mathcal{R} by removing all sort information from the signature Σ (the arity $ar(f)$ of the unsorted symbols $f \in \mathcal{F}$ becomes the number of arguments k in their rank definition(s) $f : s_1 \dots s_k \rightarrow s$), the set of variables and the rules. We let $\Theta(t)$ be the unsorted version of $t \in \mathcal{T}(\Sigma, \mathcal{X})$. In the following, we will use the set \mathcal{D} of *defined* symbols of $\Theta(\mathcal{R})$ (in the usual sense of those symbols $f \in \mathcal{F}$ having a rewrite rule

$f(l_1, \dots, l_k) \rightarrow r$ in $\Theta(\mathcal{R})$) as the set of defined symbols³ of \mathcal{R} . We have the following obvious result.

PROPOSITION 4. *Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS and $t \in \mathcal{T}(\Sigma, \mathcal{X})$ be a nonterminating term. Then, there is a minimal nonterminating term t' in t , i.e., $t \geq t'$ and $t' \in \mathcal{T}_{\Sigma, \infty}$. Furthermore, for all $t \in \mathcal{T}_{\Sigma, \infty}$, $\text{root}(\Theta(t)) \in \mathcal{D}$, where \mathcal{D} is the set of defined symbols of $\Theta(\mathcal{R})$.*

Note that terms t and t' in Proposition 4 do not need to have the same (or related) sort. Then, we have the following result, extending Proposition 2 to the order-sorted case, whose proof is analogous to that of Proposition 2, see (25).

PROPOSITION 5. *Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS and let \mathcal{D} be the set of defined symbols of $\Theta(\mathcal{R})$. For all $t \in \mathcal{T}_{\Sigma, \infty}$, there exist $l \rightarrow r \in R$, an order-sorted substitution σ and a term $u \in \mathcal{T}_{\Sigma, \infty}$ such that $\text{root}(\Theta(u)) \in \mathcal{D}$, $t \xrightarrow{\geq \Lambda}^* \sigma(l) \xrightarrow{\Lambda} \sigma(r) \geq u$ and there is a non-variable subterm v of r , $r \geq v$, such that $u = \sigma(v)$.*

3.1 Applicable rules

Given an order-sorted TRS (Σ, R) , and given a sort s , it may be the case that some rules in R can *never* be applied to any Σ -term t of sort s , nor to any term that can be obtained from t by a finite sequence of rewrites. This leads to the idea of identifying, for each sort s , a subset $\mathcal{A}_s(\mathcal{R}) \subseteq R$ of *applicable* rules, so that any rule not in $\mathcal{A}_s(\mathcal{R})$ can never be applied to a term of sort s or to any term to which such a term can eventually be rewritten. Since in general we are not assuming that the rules R are sort-decreasing, what we can always ensure about terms t' to which a term t of sort s can eventually be rewritten is that if t' has sort s' , then s and s' are in the same *connected component* in the subset ordering. That is, they belong to the same equivalence class $[s]$ for the smallest equivalence relation generated by the subsort ordering relation \leq . This leads us to the following

DEFINITION 1 (Applicable rules). *Given an OS-TRS (Σ, R) , and $s \in S$, the set $K(s)$ of s -relevant sorts is the least set satisfying:*

1. $s \in K(s)$;
2. if there is an operator declaration $f : s_1 \dots s_n \rightarrow s'$ in Σ such that $s' \in K(s)$, then $\{s_1, \dots, s_n\} \subseteq K(s)$; and
3. if $s' \in K(s)$, then $[s'] \subseteq K(s)$.

The set $\mathcal{A}_s(\mathcal{R}) \subseteq R$ of s -applicable rules is the set of rules $l \rightarrow r \in R$ such that $LS(l) \in K(s)$.

Note that under the extra assumption that \mathcal{R} is *sort-decreasing*, a more accurate set of s -applicable rules can be determined, since we need not consider all sorts in a connected component, but only certain sorts smaller than other sorts in the fixpoint computation of the set of s -relevant sorts (i.e., replace $[s']$ by $\lfloor s' \rfloor$ in Definition 1 above). In the sort-decreasing case we should not only consider the rules $l \rightarrow r$ in R , but also all the *specializations* of those rules by substitutions in which the variables are mapped to variables with smaller or equal sorts.

DEFINITION 2 (App. rules - sort dec. case). *Let (Σ, R) be a sort-decreasing OS-TRS, and s be a sort. The set $K_d(s)$ of s -relevant sorts is the least set satisfying:*

1. $s \in K_d(s)$;
2. if there is an operator declaration $f : s_1 \dots s_n \rightarrow s'$ in Σ such that $s' \in K_d(s)$, then $\{s_1, \dots, s_n\} \subseteq K_d(s)$; and
3. if $s' \in K(s)$, then $\lfloor s' \rfloor \subseteq K_d(s)$.

³This is a coarser notion than the more careful (but also less intuitive, in general) order-sorted distinction between different overloadings.

The set $\mathcal{A}_{s,d}(\mathcal{R}) \subseteq R$ (or just $\mathcal{A}_s(\mathcal{R})$ if no confusion arises) of s -applicable sort-decreasing rules is the set of rules $l \rightarrow r \in R$ such that there is a variable specialization ν such that $LS(\nu(l)) \in K_d(s)$.

EXAMPLE 3. *Consider the OS-TRS \mathcal{R} in Example 1. Note that \mathcal{R} is sort-decreasing (actually sort-preserving). The set of NzNeg-relevant sorts is $\{\text{Neg}, \text{Zero}, \text{NzNeg}\}$. Therefore, $\mathcal{A}_{\text{NzNeg},d}(\mathcal{R})$ is empty: there is no rule $l \rightarrow r$ in \mathcal{R} such that $LS(\nu(l)) \in \{\text{Neg}, \text{Zero}, \text{NzNeg}\}$ for some specialization ν . On the other hand, the set of Pos-relevant sorts is $\{\text{Pos}, \text{Zero}, \text{NzPos}, \text{Neg}, \text{NzNeg}\}$. Thus, $\mathcal{A}_{\text{Pos},d}(\mathcal{R})$ contains the following rules:*

$$\begin{aligned} \text{opposite}(p(0)) &\rightarrow s(0) \\ \text{opposite}(p(Y)) &\rightarrow s(\text{opposite}(Y)) \end{aligned}$$

where the sort of the variable Y is NzNeg.

It is then easy to show that for any term t of sort s and any rewrite sequence starting at t , all rules applied in the sequence must necessarily be in $\mathcal{A}_s(\mathcal{R})$.

PROPOSITION 6. *Let \mathcal{R} be an OS-TRS and $t, u \in \mathcal{T}(\Sigma, \mathcal{X})$ be such that t is of sort s . Then, $t \xrightarrow{*}_{\mathcal{R}} u$ if and only if $t \xrightarrow{*}_{\mathcal{A}_s(\mathcal{R})} u$. Furthermore, if \mathcal{R} is sort-decreasing, then $t \xrightarrow{*}_{\mathcal{R}} u$ if and only if $t \xrightarrow{*}_{\mathcal{A}_{s,d}(\mathcal{R})} u$.*

In the following, we often generically use $\mathcal{A}_s(\mathcal{R})$; the particular definition behind this notation (Definition 1 or 2 above) will be clarified from the context.

3.2 Order-sorted dependency pairs

We need to properly extend the order-sorted signature Σ to a *marked* order-sorted signature Σ^\sharp . The set of sorts S is enriched with a new fresh sort symbol \top^\sharp unrelated to other sorts. We extend Σ to Σ^\sharp by adding, for each declaration $f : \vec{s} \rightarrow s$ in Σ , a new declaration $f^\sharp : \vec{s} \rightarrow \top^\sharp$. This is a technical trick to have all marked symbols having the same (output) sort; hence the left- and right-hand sides of the OS-DPs will also have the same sort and can be considered as (sort-preserving) rules. Since marked symbols are only reduced at top positions this does not yield any problem. Denote by $\mathcal{T}^\sharp(\Sigma, \mathcal{X}) \subseteq \mathcal{T}(\Sigma^\sharp, \mathcal{X})$ the set of *marked* order-sorted terms: $\mathcal{T}^\sharp(\Sigma, \mathcal{X}) = \cup_{s \in S} \{t^\sharp \mid t \in \mathcal{T}(\Sigma, \mathcal{X})_s - \mathcal{X}_s\}$, where $t^\sharp = f^\sharp(t_1, \dots, t_k)$ if $t = f(t_1, \dots, t_k)$ (conversely, $t^\sharp = f(t_1, \dots, t_k)$ if $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is the unmarked version of $t = f^\sharp(t_1, \dots, t_k) \in \mathcal{T}^\sharp(\Sigma, \mathcal{X})$). We can give a first definition of order-sorted dependency pairs.

DEFINITION 3 (Basic OS-DPs). *Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS. We define the set $\text{BOSDP}(\mathcal{R})$ of basic order-sorted dependency pairs (BOS-DPs) as follows⁴:*

$$\text{BOSDP}(\mathcal{R}) = \{l^\sharp \rightarrow v^\sharp \mid l \rightarrow r \in R, r \geq v, \text{root}(v) \in \mathcal{D}, l \not\geq v\}$$

Note that pairs $u \rightarrow v \in \text{BOSDP}(\mathcal{R})$ are *order-sorted* rules and $u, v \in \mathcal{T}^\sharp(\Sigma, \mathcal{X})_{\top^\sharp}$. Although, from a theoretical point of view, Definition 3 leads to a correct dependency pairs approach for OS-TRSs (see Theorem 1 below), some useless pairs could be generated (see the examples below). Thus, it makes sense to use a combination of (unsorted) syntactic checking and a reducibility test by *narrowing*. Actually, this is implicit in Proposition 5: if a nonterminating sequence must be issued from a minimal nonterminating term, then the nonvariable part v of the right-hand side r of the rule which is a prefix of such a term must be eventually narrowable

⁴The restriction $l \not\geq v$ follows from (9), see also (25).

(within r). Our definition of order-sorted dependency pair uses a function REN defined as follows:

$$\text{REN}(x) = \begin{cases} x & \text{if } x \in \mathcal{X}_s \text{ and } \mathcal{A}_s(\mathcal{R}) = \emptyset \\ y & \text{if } x \in \mathcal{X}_s, y \in \mathcal{X}_s, \text{ and } \mathcal{A}_s(\mathcal{R}) \neq \emptyset \end{cases}$$

$$\text{REN}(f(t_1, \dots, t_k)) = \begin{cases} f(t_1, \dots, t_k) & \\ \text{if } s = \text{LS}(f(t_1, \dots, t_k)) \text{ and } \mathcal{A}_s(\mathcal{R}) = \emptyset \\ f(\text{REN}(t_1), \dots, \text{REN}(t_k)) & \text{otherwise} \end{cases}$$

where y is intended to be a fresh new variable of sort s which has not yet been used (we could think of y as the ‘next’ variable in an infinite list of variables of the appropriate sort). This leads to the following.

DEFINITION 4 (OS-DPs). Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS. We define the set $\text{OSDP}(\mathcal{R})$ of order-sorted dependency pairs (OS-DPs) as follows:

$$\text{OSDP}(\mathcal{R}) = \{l^\sharp \rightarrow v^\sharp \mid l \rightarrow r \in R, \exists p \in \text{Pos}(r), v = r|_p, \text{root}(v) \in \mathcal{D}, \exists r', \text{REN}(r) \xrightarrow{q} r', p \leq q, l \not\prec v\}$$

Let us illustrate the previous definitions with some examples.

EXAMPLE 4. The following MS-TRS, which corresponds to the famous Toyama’s example, is a modified version of one in (43, Section 3.3)

```
fmod Toyama-MS is
  sorts S1 S2 S .
  op 0 : -> S1 .
  op 1 : -> S1 .
  op f : S1 S1 S1 -> S .
  op g : S2 S2 -> S2 .
  vars x : S1 .
  vars y z : S2 .
  eq f(0,1,x) = f(x,x,x) .
  eq g(y,z) = y .
  eq g(y,z) = z .
endfm
```

Viewed as a TRS, the system is nonterminating. According to Theorem 2 below, we can prove its termination (as a special case of OS-TRS) by noticing that there is no OS-DP: since $\mathcal{A}_{S1}(\mathcal{R})$ is empty, $\text{REN}(f(x,x,x)) = f(x,x,x)$. Since $f(x,x,x)$ is not narrowable, $\text{OSDP}(\mathcal{R}) = \emptyset$. In contrast, $\text{BOSDP}(\mathcal{R}) = \{F(0,1,x) \rightarrow F(x,x,x)\}$.

EXAMPLE 5. For **EVEN** in Figure 1, BOS-DPs and OS-DPs coincide:

- (1) $\text{IS-EVEN}(s(s(X))) \rightarrow \text{IS-EVEN}(X)$
- (2) $\text{IS-EVEN}(Y) \rightarrow \text{IS-EVEN}(\text{opposite}(Y))$
- (3) $\text{IS-EVEN}(Y) \rightarrow \text{OPPOSITE}(Y)$
- (4) $\text{OPPOSITE}(p(Y)) \rightarrow \text{OPPOSITE}(Y)$

where variable X has sort Pos and variable Y has sort NzNeg .

4. Order-sorted dependency chains

A central notion in the dependency pairs approach is that of *chain* (5; 20). Chains are finite or infinite sequences $u_i \rightarrow v_i$ for $i \geq 1$ of pairs satisfying $\text{Var}(u_i) \cap \text{Var}(u_j) = \emptyset$ for all $i \neq j$ which are connected as follows: there is a substitution σ such that, for each $i \geq 1$ we have $\sigma(v_i) \rightarrow_{\mathcal{R}}^* \sigma(u_{i+1})$. We adapt this notion to the OS case in our next definition. Let $\mathcal{P}^\sharp(\Sigma, \mathcal{X})$ be the S -sorted set of pairs of marked order sorted terms: $\mathcal{P}^\sharp(\Sigma, \mathcal{X}) = \bigcup_{s \in S} \{\mathcal{T}^\sharp(\Sigma, \mathcal{X})_s \times \mathcal{T}^\sharp(\Sigma, \mathcal{X})_s\}$.

DEFINITION 5 (Chain). Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS. Given $\mathcal{P} \subseteq \mathcal{P}^\sharp(\Sigma, \mathcal{X})$, an $(\mathcal{R}, \mathcal{P})$ -chain is a finite or infinite sequence of

pairs $u_i \rightarrow v_i \in \mathcal{P}$, for $i \geq 1$ (as usual, we assume that $\text{Var}(v_i) \cap \text{Var}(u_j) = \emptyset$ for all $i \neq j$; renamings are used if necessary) such that there is an order-sorted substitution σ satisfying $\sigma(v_i) \rightarrow_{\mathcal{R}}^* \sigma(u_{i+1})$ for all $i \geq 1$.

An $(\mathcal{R}, \mathcal{P})$ -chain is called *minimal* if for all $i \geq 1$ and all t such that $\sigma(v_i) \geq t$, t is \mathcal{R} -terminating. If \mathcal{R} is sort-decreasing, then according to Proposition 3 we can just require that (as usual) for all $i \geq 1$, $\sigma(v_i)$ is \mathcal{R} -terminating.

In the following, we establish the correctness of the (basic) order-sorted dependency pairs approach(es).

THEOREM 1 (Correctness of BOS-DPs). Let \mathcal{R} be an OS-TRS. If there is no infinite minimal $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain, then \mathcal{R} is terminating.

PROOF. By contradiction. If $\mathcal{R} = (\Sigma, R)$ is not terminating, then there is a minimal nonterminating term $t \in \mathcal{T}_{\Sigma, \infty}$. By Proposition 5, there is a rule $l \rightarrow r \in R$, an order-sorted substitution σ and a term $u \in \mathcal{T}_{\Sigma, \infty}$ such that $t \xrightarrow{\geq \Delta^*} \sigma(l) \xrightarrow{\Delta} \sigma(r) \geq u$ and there is a subterm v of r such that $u = \sigma(v)$. Since $u \in \mathcal{T}_{\Sigma, \infty}$, by using Proposition 5 again, we conclude that there is a rule $\lambda \rightarrow \rho$

such that $u = \sigma(v) \xrightarrow{\geq \Delta^*} \sigma(\lambda)$ (since we can assume that the variables in this rule do not occur in l , we can use the same – conveniently extended – substitution σ). According to Definition 3, we have a dependency pair $l^\sharp \rightarrow v^\sharp \in \text{BOSDP}(\mathcal{R})$ such that $u = \sigma(v) \in \mathcal{T}_{\Sigma, \infty}$, i.e., we can start an $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain beginning with $\sigma(l^\sharp) \rightarrow_{\text{BOSDP}(\mathcal{R})} \sigma(v^\sharp)$. Since $\sigma(v) \in \mathcal{T}_{\Sigma, \infty}$, $\sigma(v^\sharp)$ is \mathcal{R} -terminating and all its proper subterms also are. Hence, $u^\sharp \rightarrow_{\mathcal{R}}^* \sigma(\lambda^\sharp)$ and we can start an infinite minimal $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain, which could be infinitely extended in a similar way by starting from u^\sharp now to obtain an infinite minimal $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain. This contradicts our initial assumption. \square

The following example shows that Theorem 1 does *not* hold for OS-DPs.

EXAMPLE 6. Consider the following nonterminating OS-TRS:

```
fmod EXAMPLE_6 is
  sorts S1 S2 S .
  subsorts S1 < S .
  subsorts S2 < S .
  op a : -> S1 .
  op b : -> S2 .
  op f : S1 -> S1 .
  op g : S -> S .
  var X : S1 .
  eq f(X) = g(X) .
  eq g(b) = f(a) .
  eq a = b .
endfm
```

Since $\text{REN}(g(X)) = g(Y)$ for some variable Y of sort $S1$, $\text{REN}(g(X))$ cannot be narrowed ($g(Y)$ and $g(b)$ do not have any OS-unifier). Thus, there is only one OS-DP: $G(b) \rightarrow F(a)$ which does not induce any chain. We would wrongly conclude termination of \mathcal{R} . In contrast, there are two BOS-DPs: (1) $F(X) \rightarrow G(X)$ and (2) $G(b) \rightarrow F(a)$ which induce an infinite chain corresponding to the nonterminating behavior of \mathcal{R} :

$$\underline{F(a)} \rightarrow_{\text{OSDP}(\mathcal{R})} \underline{G(a)} \rightarrow_{\mathcal{R}} \underline{G(b)} \rightarrow_{\text{OSDP}(\mathcal{R})} \underline{F(a)} \rightarrow \dots$$

With sort-decreasing OS-TRSs, we can use the more accurate set of OS-DPs.

THEOREM 2 (Correctness of OS-DPs). Let \mathcal{R} be a sort-decreasing OS-TRS. If there is no infinite minimal $(\mathcal{R}, \text{OSDP}(\mathcal{R}))$ -chain, then \mathcal{R} is terminating.

PROOF. By contradiction. Consider u, v as in the proof of Theorem 1. It is not difficult to see that $\text{REN}(v)$ can be *narrowed*. There are two cases:

1. If all rewriting steps in the sequence from $\sigma(v)$ to $\sigma(\lambda)$ occur on the bindings for the variables in v (i.e., *below* the occurrences of function symbols in v but possibly in an independent way depending on the particular occurrences of the variables in v), then $\sigma(\lambda) = \theta(\text{REN}(v))$ for some substitution θ . Here:
 - (a) the *renaming* of ‘potentially reducible’ (bindings for) variables of v which is obtained by using REN is essential to keep all reductions on the original bindings $\sigma(x)$ completely independent.
 - (b) sort-decreasingness ensures that the sort of the reducts is smaller than or equal to the sort of the corresponding variables introduced by REN . Thus, they can be used as bindings for such variables.

Therefore, $\text{REN}(v)$ can be narrowed by using the rule $\lambda \rightarrow \rho$.

2. There is a sequence $\sigma(v) \xrightarrow{*} \theta(\text{REN}(v)) \rightarrow u' \xrightarrow{*} \sigma(\lambda)$ for some substitution θ and the rewrite step on $\theta(\text{REN}(v))$ is issued at a position $p \in \mathcal{P}_{\text{OS}\Sigma}(\text{REN}(v)) = \mathcal{P}_{\text{OS}\Sigma}(v)$ by using some rule $l' \rightarrow r'$. Thus, $\text{REN}(v)$ can also be narrowed by using the rule $l' \rightarrow r'$.

According to Definition 4, we have a dependency pair $l^\sharp \rightarrow v^\sharp \in \text{OSDP}(\mathcal{R})$ such that $u = \sigma(v) \in \mathcal{T}_{\Sigma, \infty}$, i.e., we can start an $(\mathcal{R}, \text{OSDP}(\mathcal{R}))$ -chain beginning with $\sigma(l^\sharp) \xrightarrow{\text{OSDP}(\mathcal{R})} \sigma(v^\sharp)$, where $\sigma(v^\sharp)$ is \mathcal{R} -terminating. Hence, $u^\sharp \xrightarrow{*}_{\mathcal{R}} \sigma(\lambda^\sharp)$ and we can start an $(\mathcal{R}, \text{OSDP}(\mathcal{R}))$ -chain, which could be infinitely extended in a similar way by starting from u^\sharp now to obtain an infinite minimal $(\mathcal{R}, \text{OSDP}(\mathcal{R}))$ -chain. This contradicts our initial assumption. \square

Fortunately, sort-decreasingness is not a very strong restriction in practice. In particular, many-sorted TRSs are sort-decreasing by definition. And sort-decreasingness is usually required (for semantical reasons) for the practical use of OS-TRSs (23; 30), also for termination analysis (21; 40). Now we prove that the previous OS-dependency pairs approach is not only correct but also complete for proving termination of *sort-decreasing* OS-TRSs.

THEOREM 3 (Completeness). *Let \mathcal{R} be a sort-decreasing OS-TRS. If \mathcal{R} is terminating, then there is no infinite $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain.*

PROOF. By contradiction. If there is an infinite $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain, then there is an order-sorted substitution σ and dependency pairs $u_i \rightarrow v_i \in \text{BOSDP}(\mathcal{R})$ such that $\sigma(v_i) \xrightarrow{*}_{\mathcal{R}} \sigma(u_{i+1})$, for $i \geq 1$. Now, consider the first dependency pair $u_1 \rightarrow v_1$ in the sequence: Then, v_1^\sharp is a subterm of the right-hand-side r_1 of a rule $l_1 \rightarrow r_1$ in \mathcal{R} , where $LS(l_1)$ and $LS(r_1)$ (hence $LS(\sigma(l_1))$ and $LS(\sigma(r_1))$) are in the same connected component of (S, \leq) . Therefore, $r_1 = C_1[v_1^\sharp]_{p_1}$ for some $p_1 \in \mathcal{P}_{\text{OS}\Sigma}(r_1)$ and we can perform the order-sorted rewriting step $t_1 = \sigma(u_1) \xrightarrow{\mathcal{R}} \sigma(r_1) = \sigma(C_1)[\sigma(v_1^\sharp)]_{p_1} = s_1$, where $\sigma(v_1^\sharp)^\sharp = \sigma(v_1) \xrightarrow{*}_{\mathcal{R}} \sigma(u_2)$ and $\sigma(u_2)$ initiates an infinite $(\mathcal{R}, \text{BOSDP}(\mathcal{R}))$ -chain. By sort-decreasingness of \mathcal{R} , $\sigma(C_1)[u]$ is well-formed for all reducts u of $\sigma(v_1)$ in the rewrite sequence $\sigma(v_1) \xrightarrow{*}_{\mathcal{R}} \sigma(u_2)$. Thus, we can actually say that $\sigma(C_1[v_1]) \xrightarrow{*}_{\mathcal{R}} \sigma(C_1[u_2])$. Hence, we can build in this way an infinite order-sorted rewrite sequence $t_1 \xrightarrow{\mathcal{R}} s_1 \xrightarrow{*}_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \dots$, which contradicts termination of \mathcal{R} . \square

Sort-decreasingness cannot be removed from Theorem 3.

EXAMPLE 7. Consider the following OS-TRS:

```
fmod EXAMPLE_7 is
  sorts S1 S2 S .
  subsorts S1 < S .
  subsorts S2 < S .
  op a : -> S1 .
  op c : S1 -> S2 .
  eq a = c(a) .
endfm
```

which is not sort-decreasing. There is a BOS-DP: $A \rightarrow A$ which induces an infinite chain $A \rightarrow A \rightarrow \dots$. However, this OS-TRS is clearly terminating: an attempt to reduce $c(a)$ would yield the ill-formed term $c(c(a))$. Note that $A \rightarrow A$ is an OS-DP as well.

Since $\text{OSDP}(\mathcal{R}) \subseteq \text{BOSDP}(\mathcal{R})$, we have the following corollary of Theorem 3.

COROLLARY 1. *Let \mathcal{R} be a sort-decreasing OS-TRS. If \mathcal{R} is terminating, then there is no infinite $(\mathcal{R}, \text{OSDP}(\mathcal{R}))$ -chain.*

5. Order-sorted dependency graph

As noticed by Arts and Giesl (5), the analysis of infinite chains of DPs can be performed by looking at (the cycles \mathcal{C} of) the *dependency graph* $\text{DG}(\mathcal{R})$ associated to the TRS \mathcal{R} . The nodes of the dependency graph are the dependency pairs in $\text{DP}(\mathcal{R})$; there is an arc from a dependency pair $u \rightarrow v$ to a dependency pair $u' \rightarrow v'$ (where $\text{Var}(v) \cap \text{Var}(u') = \emptyset$) if there is a substitution σ such that $\sigma(v) \xrightarrow{*}_{\mathcal{R}} \sigma(u')$. In practice, it is better to abstract the dependency graph from the dependency pairs $\text{DP}(\mathcal{R})$ of a TRS \mathcal{R} and think of a dependency graph $\text{DG}(\mathcal{R}, \mathcal{P})$ associated to a set of pairs \mathcal{P} and a TRS \mathcal{R} (20): we use now \mathcal{P} (instead of $\text{DP}(\mathcal{R})$) as the set of nodes, as indicated above (20, Definition 7). Furthermore, in practice, we are concerned with the so-called *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) (25; 26). A strongly connected component in the OS-DG is a *maximal cycle*, i.e., it is not contained in any other cycle.

In order-sorted rewriting we have to instantiate (dependency) pairs by using *order-sorted* substitutions; we then connect them by using order-sorted rewriting. This leads to the following definition of order-sorted dependency graph:

DEFINITION 6 (Order-Sorted Dependency Graph). *Let \mathcal{R} be an OS-TRS. The order-sorted dependency graph (OS-DG) consists of the set $\text{OSDP}(\mathcal{R})$ of OS-DPs together with arcs which connect them as follows: There is an arc from $u \rightarrow v \in \text{OSDP}(\mathcal{R})$ to $u' \rightarrow v' \in \text{OSDP}(\mathcal{R})$ (where $\text{Var}(v) \cap \text{Var}(u') = \emptyset$) if there is an order-sorted substitution σ such that $\sigma(v) \xrightarrow{*}_{\mathcal{R}} \sigma(u')$.*

Using $\text{BOSDP}(\mathcal{R})$ instead of $\text{OSDP}(\mathcal{R})$, we would have the analogous notion of BOS-DG. Again, by Proposition 6, we could use $\rightarrow_{\mathcal{A}_s(\mathcal{R})}$ (or $\rightarrow_{\mathcal{A}_s, d(\mathcal{R})}$ for sort-decreasing OS-TRSs \mathcal{R}) instead of $\rightarrow_{\mathcal{R}}$ in Definition 6 (for $s = LS(v)$). Given an OS-TRS $\mathcal{R} = (\Sigma, R)$ and $\mathcal{P} \subseteq \mathcal{P}^\sharp(\Sigma, \mathcal{X})$, the notion of an order-sorted dependency graph $\text{OSDG}(\mathcal{R}, \mathcal{P})$ associated to \mathcal{R} and \mathcal{P} would be given in a similar way as for the unsorted case. The following result (whose proof is obvious) formalizes the use of the dependency graph (see also (20, Theorem 8)).

THEOREM 4 (Use of SCCs). *Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS and $\mathcal{P} \subseteq \mathcal{P}^\sharp(\Sigma, \mathcal{X})$ be finite. There is an infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain A if and only if there is an SCC \mathcal{C} in $\text{OSDG}(\mathcal{R}, \mathcal{P})$ containing a cycle \mathcal{C} such that A contains a tail B which is an infinite minimal $(\mathcal{R}, \mathcal{C})$ -chain.*

In general, $\text{OSDG}(\mathcal{R}, \mathcal{P})$ is *not* computable, and we need to approximate it. The main idea is to *approximate* rewriting from $\sigma(v)$ to $\sigma(u')$ as *unification* of u' and an abstraction of v (5). Function CAP is used to abstract v by replacing possibly reducible subterms of v by fresh variables which represent possible independent reductions on v . Let CAP be given as follows:

$$\text{CAP}(x) = x \text{ if } x \text{ is a variable}$$

$$\text{CAP}(f(t_1, \dots, t_k)) = \begin{cases} f(t_1, \dots, t_k) & \text{if } s = \text{LS}(f(t_1, \dots, t_k)) \\ & \text{and } \mathcal{A}_s(\mathcal{R}) = \emptyset \\ y & \text{if } s = \text{LS}(f(t_1, \dots, t_k)), f \in \mathcal{D}, \\ & y \in \mathcal{X}_s, \text{ and } \mathcal{A}_s(\mathcal{R}) \neq \emptyset, \\ f(\text{CAP}(t_1), \dots, \text{CAP}(t_k)) & \text{otherwise} \end{cases}$$

As for REN above, y is intended to be a fresh new variable of sort s which has not yet been used. Then, we have an arc from $u \rightarrow v$ to $u' \rightarrow v'$ if $\text{REN}(\text{CAP}(v))$ and u' unify by means of an *order-sorted unifier*. Following (5), we say that v and u' are *OS-connectable*. The following result formalizes this approach.

PROPOSITION 7. *Let \mathcal{R} be a sort-decreasing OS-TRS. If there is an arc from $u \rightarrow v$ to $u' \rightarrow v'$ in the OS-DG, then v and u' are OS-connectable.*

The following example shows that Proposition 7 does not hold if sort-decreasingness is not required.

EXAMPLE 8. *Consider the OS-TRS \mathcal{R} and its BOS-DPs (1) and (2) as in Example 6. Since $\text{REN}(\text{CAP}(\mathbf{G}(\mathbf{X}))) = \mathbf{G}(\mathbf{Y})$ (for some new variable \mathbf{Y} of sort $\mathbf{S1}$) and $\mathbf{G}(\mathbf{b})$ do not unify (because the sort of \mathbf{b} is $\mathbf{S2}$), (1) and (2) are not OS-connectable. However, there is an arc from (1) to (2) in the BOS-DG.*

The *estimated* dependency graph $\text{EOSDG}(\mathcal{R}, \mathcal{P})$ is the graph built using the pairs in \mathcal{P} as nodes and drawing an arc from $u \rightarrow v$ to $u' \rightarrow v'$ if v and u' are OS-connectable (using \mathcal{R}). The following example shows that a careful definition of the OS-DPs, as discussed in Section 3, is important for the definition and practical use of the EOS-DG.

EXAMPLE 9. *Consider the following terminating OS-TRS:*

```
fmod EXAMPLE_9 is
  sorts S S' .
  subsorts S' < S .
  op a : -> S .
  op f : S -> S .
  op f : S' -> S .
  var X : S' .
  eq f(X) = f(f(a)) .
endfm
```

It has no OS-DPs ($\mathbf{f}(\mathbf{f}(\mathbf{a}))$ is not narrowable). In contrast, it has two BOS-DPs: $\mathbf{F}(\mathbf{X}) \rightarrow \mathbf{F}(\mathbf{f}(\mathbf{a}))$ and $\mathbf{F}(\mathbf{X}) \rightarrow \mathbf{F}(\mathbf{a})$. In the estimated dependency graph we would connect the first pair to itself: $\text{REN}(\text{CAP}(\mathbf{F}(\mathbf{f}(\mathbf{a})))) = \mathbf{F}(\mathbf{Y})$ (where the rank of \mathbf{F} is $\mathbf{S} \rightarrow \top^\sharp$ and \mathbf{Y} has sort \mathbf{S}) and $\mathbf{F}(\mathbf{X})$ (where the rank of \mathbf{F} is $\mathbf{S}' \rightarrow \top^\sharp$ and \mathbf{X} has sort \mathbf{S}') unify by using the OS-substitution $\sigma(\mathbf{Y}) = \mathbf{X}$. In contrast, the second dependency pair cannot be connected either with itself or with the first one: $\text{REN}(\text{CAP}(\mathbf{F}(\mathbf{a})))) = \mathbf{F}(\mathbf{a})$ and $\mathbf{F}(\mathbf{X})$ do not unify (because \mathbf{X} , whose sort is \mathbf{S}' , cannot be instantiated to \mathbf{a} , whose sort is \mathbf{S} , due to $\mathbf{S}' < \mathbf{S}$).

EXAMPLE 10. *(Continuing Example 5) For the OS-TRS EVEN in Figure 1 with (B)OS-DPs as in Example 5, we have the EOS-DG in Figure 3. Note that there is no arc from (1) to (2) (nor (3)): $\text{IS-EVEN}(\mathbf{Y})$ and $\text{IS-EVEN}(\mathbf{X})$ do not unify because \mathbf{X} has sort Pos , \mathbf{Y} has sort NzNeg and there is no sort s such that $s \leq \text{Pos}$*

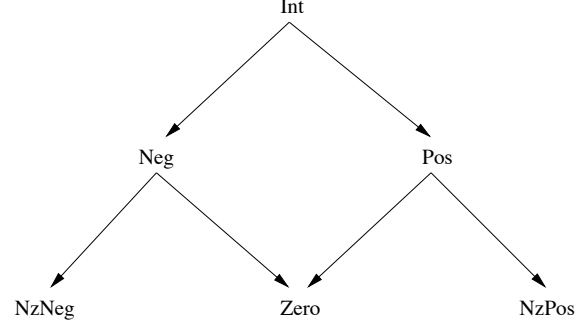


Figure 2. Ordering among sorts for the OS-TRS EVEN

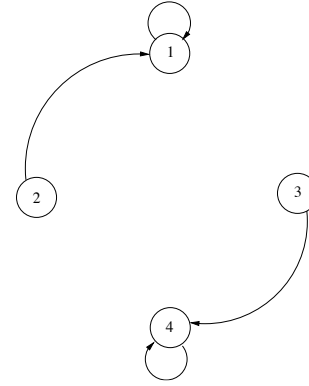


Figure 3. OS-DG for the OS-TRS EVEN

and $s \leq \text{NzNeg}$. And there is no arc from (2) to (2) (nor (3)): $\text{REN}(\text{CAP}(\text{IS-EVEN}(\text{opposite}(\mathbf{Y}))))$ is $\text{IS-EVEN}(\mathbf{Z})$ for some variable \mathbf{Z} of sort NzPos . But $\text{IS-EVEN}(\mathbf{Y})$ and $\text{IS-EVEN}(\mathbf{Z})$ do not unify because \mathbf{Y} has sort NzNeg , \mathbf{Z} has sort NzPos and there is no sort s such that $s \leq \text{NzNeg}$ and $s \leq \text{NzPos}$. Thus, the EOS-DG has two cycles: $\mathcal{C}_1 = \{(1)\}$ and $\mathcal{C}_2 = \{(4)\}$.

5.1 Proving termination of OS-TRSs using the dependency graph

When proving termination of a sort-decreasing OS-TRS \mathcal{R} , the starting point is Theorem 2 which says that \mathcal{R} is terminating if there is no infinite minimal $(\mathcal{R}, \text{OSDP}(\mathcal{R}))$ -chain. Theorem 4 allows us to focus our attention on the SCCs of the (estimated) OS-DG: we have to prove the absence of infinite minimal $(\mathcal{R}, \mathcal{C})$ -chains for all SCCs \mathcal{C} in the graph. As in (20), the main idea is the following: use orderings among terms to remove pairs from a set of pairs \mathcal{P} (initially an SCC in the order-sorted dependency graph $\text{OSDG}(\mathcal{R})$ of the OS-TRS \mathcal{R}), thus obtaining a new (strict) subset of pairs $\mathcal{P}' \subset \mathcal{P}$ such that if there is an infinite $(\mathcal{R}, \mathcal{P})$ -chain, then there is an infinite $(\mathcal{R}, \mathcal{P}')$ -chain. Now we can use Theorem 4 again to obtain a new set of problems for each SCC in $\text{OSDG}(\mathcal{R}, \mathcal{P}')$ to which we can apply the same procedure. We obtain a finite tree of ‘problems’ to solve. The nodes of this tree are pairs $(\mathcal{R}, \mathcal{P})$. If all leaves of the tree have an empty ‘pair component’ (i.e., $\mathcal{P} = \emptyset$), then we conclude termination of \mathcal{R} .

REMARK 2. *The procedure sketched above corresponds to the use of DP-processors in the dependency pair framework (20).*

In the following sections we discuss how to use orderings among terms for this purpose.

6. Use of reduction pairs

A reduction pair (\succeq, \sqsupset) consists of a stable and monotonic quasi-ordering \succeq on (unsorted) terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and a stable and well-founded ordering \sqsupset (again on $\mathcal{T}(\mathcal{F}, \mathcal{X})$) satisfying $\succeq \circ \sqsupset \subseteq \sqsupset \circ \succeq$ or $\sqsupset \circ \succeq \subseteq \sqsupset \circ \succeq$ (31; 15). Recall that a *quasi-ordering* is a reflexive and transitive relation (on terms); by an *ordering* we mean an irreflexive and transitive relation. Here, stable means that, for all substitution σ and terms $t, u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $\sigma(t) \succeq \sigma(u)$ (resp. $\sigma(t) \sqsupset \sigma(u)$) whenever $t \succeq u$ (resp. $t \sqsupset u$); *monotonic* means that for all k -ary symbols $f \in \mathcal{F}$, terms $t, u, t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $i, 1 \leq i \leq k$, we have that $f(t_1, \dots, t_{i-1}, t, \dots, t_k) \succeq f(t_1, \dots, t_{i-1}, u, \dots, t_k)$ whenever $t \succeq u$.

Given a TRS \mathcal{R} and a set of pairs \mathcal{P} , the quasi-ordering \succeq is intended to be used to compare the left- and right-hand sides of the rules $l \rightarrow r \in \mathcal{R}$, and the strict ordering \sqsupset is used to compare the two components of the pairs $u \rightarrow v \in \mathcal{P}$. In both cases, it is possible to use some *argument filtering* before doing the comparisons (31; 5).

In this section we consider the most obvious way to deal with order-sorted dependency chains: since order-sorted rewriting steps with an OS-TRS \mathcal{R} are always possible in the *underlying* (unsorted) TRS $\Theta(\mathcal{R})$, we can just forget all sort information in the cycles \mathcal{C} and applicable rules $\mathcal{A}_{\mathcal{C}}(\mathcal{R})$ for each cycle and use standard reduction pairs on them.

DEFINITION 7. Given an OS-TRS $\mathcal{R} = (\Sigma, R)$ and $\mathcal{P} \subseteq \mathcal{P}^\#(\Sigma, \mathcal{X})$, the set of \mathcal{P} -applicable rules $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$ is:

$$\mathcal{A}_{\mathcal{P}}(\mathcal{R}) = \bigcup_{u \rightarrow f^\#(v_1, \dots, v_k) \in \mathcal{P}} \bigcup_{1 \leq i \leq k} \mathcal{A}_{LS(v_i)}(\mathcal{R}).$$

Actually, we can further generalize this point of view: let Γ be a transformation mapping an order-sorted signature Σ into an unsorted signature \mathcal{F}_Γ , the set $\mathcal{T}(\Sigma, \mathcal{X})$ into $\mathcal{T}(\mathcal{F}_\Gamma, \mathcal{X})$ and an OS-TRS \mathcal{R} over Σ into a TRS $\Gamma(\mathcal{R})$ over \mathcal{F}_Γ in such a way that for all $t, u \in \mathcal{T}(\Sigma, \mathcal{X})$, $t \rightarrow_{\mathcal{R}}^* u$ implies $\Gamma(t) \rightarrow_{\Gamma(\mathcal{R})}^* \Gamma(u)$. Then, the following result easily follows:

THEOREM 5. Let $\mathcal{R} = (\Sigma, R)$ be an OS-TRS and $\mathcal{P} \subseteq \mathcal{P}^\#(\Sigma, \mathcal{X})$ be finite. Let (\succsim, \sqsupset) be a reduction pair such that, $\Gamma(\mathcal{A}_{\mathcal{P}}(\mathcal{R})) \subseteq \succsim$, $\Gamma(\mathcal{P}) \subseteq \succsim \cup \sqsupset$, and $\Gamma(\mathcal{P}) \cap \sqsupset \neq \emptyset$. Let $\mathcal{P}_{\sqsupset} = \{u \rightarrow v \in \mathcal{P} \mid \Gamma(u) \sqsupset \Gamma(v)\}$. There is an infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain if and only if there is an infinite minimal $(\mathcal{R}, \mathcal{P} - \mathcal{P}_{\sqsupset})$ -chain.

PROOF. The if part is obvious. For the *only if* part, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain A , but that there is no infinite minimal $(\mathcal{R}, \mathcal{P} - \mathcal{P}_{\sqsupset})$ -chain. Due to the finiteness of \mathcal{P} , we can assume that there is $\mathcal{P}' \subseteq \mathcal{P}$ such that A has a tail B

$$t_1 \rightarrow_{\mathcal{R}}^* t'_1 \rightarrow_{\mathcal{P}'} t_2 \rightarrow_{\mathcal{R}}^* t'_2 \rightarrow_{\mathcal{P}'} \dots,$$

where all pairs in \mathcal{P}' are infinitely often used and t_i is \mathcal{R} -terminating for all $i \geq 1$ (by minimality of A). No pair $u \rightarrow v \in \mathcal{P}'$ satisfies that $\Gamma(u) \sqsupset \Gamma(v)$. Otherwise, by monotonicity and stability of \succsim and by using the fact that $\Gamma(\mathcal{A}_{\mathcal{P}}(\mathcal{R})) \subseteq \succsim$, we would have that $\Gamma(t_i) \succsim \Gamma(t'_i)$ for all $i \geq 1$. By stability of \sqsupset (and of \succsim), we have that $\Gamma(t'_i) (\succsim \cup \sqsupset) \Gamma(t_{i+1})$ for all $i \geq 1$ and that $\Gamma(t'_j) \sqsupset \Gamma(t_{j+1})$ for all $j \in J$ for an infinite set $J = \{j_1, \dots, j_n, \dots\}$ of natural numbers $j_1 < j_2 < \dots < j_n < \dots$. Now, since $\succsim \circ \sqsupset \subseteq \sqsupset \circ \succsim$ or $\sqsupset \circ \succsim \subseteq \sqsupset \circ \succsim$, we would obtain an infinite sequence

$$\Gamma(t_{j_1}) \sqsupset \Gamma(t_{j_2}) \sqsupset \Gamma(t_{j_3}) \sqsupset \dots,$$

which contradicts well-foundedness of \sqsupset . Therefore, $\mathcal{P}' \subseteq (\mathcal{P} - \mathcal{P}_{\sqsupset})$, which means that B is an infinite minimal $(\mathcal{R}, \mathcal{P} - \mathcal{P}_{\sqsupset})$ -chain, thus leading to a contradiction. \square

We can use the transformation Θ which removes all sort information from the symbols (hence from the terms and rules) and yields the *underlying* TRS $\Theta(\mathcal{R})$.

EXAMPLE 11. Consider the OS-TRS EVEN (here denoted \mathcal{R}) in Figure 1 and the SCCs \mathcal{C}_1 and \mathcal{C}_2 in the EOS-DG as shown in Example 10. According to Example 3, we have that $\mathcal{A}_{\mathcal{C}_2}(\mathcal{R})$ is empty and $\mathcal{A}_{\mathcal{C}_1}(\mathcal{R})$ consists of the two rules for function **opposite**. The reduction pair $(=, >_{rpo})$ (for any $RPO^5 >_{rpo}$) can be used to show that there is no infinite $(\Theta(\mathcal{A}_{\mathcal{C}_2}(\mathcal{R})), \Theta(\mathcal{C}_2))$ -chain: since $\text{OPPOSITE}(\mathbf{p}(\mathbf{Y})) >_{rpo} \text{OPPOSITE}(\mathbf{Y})$, by Theorem 5 this is equivalent to the existence of infinite (\mathcal{R}, \emptyset) -chains. But there is no (\mathcal{R}, \emptyset) -chain at all. On the other hand, the reduction pair $(>_{rpo}, >_{rpo})$ consisting of the RPO induced by the precedence **opposite** $\succ \mathbf{s}$ can be used to show that there is no infinite $(\Theta(\mathcal{A}_{\mathcal{C}_1}(\mathcal{R})), \Theta(\mathcal{C}_1))$ -chain in a similar way. Thus, \mathcal{R} is terminating.

Of course, this ‘forgetful’ technique often fails to achieve termination proofs.

EXAMPLE 12. Consider the OS-TRS FACTORIAL in Figure 1. Viewed as a TRS, this system is obviously nonterminating (due to the second rule for **fact**). We have the following OS-DPs:

- (1) $\mathbf{x} \ +^\# \ \mathbf{s}(\mathbf{y}) \ \rightarrow \ \mathbf{x} \ +^\# \ \mathbf{y}$
- (2) $\mathbf{x} \ *^\# \ \mathbf{s}(\mathbf{y}) \ \rightarrow \ \mathbf{x} \ +^\# \ (\mathbf{x} \ * \ \mathbf{y})$
- (3) $\mathbf{x} \ *^\# \ \mathbf{s}(\mathbf{y}) \ \rightarrow \ \mathbf{x} \ *^\# \ \mathbf{y}$
- (4) $\text{FACT}(\mathbf{x}') \ \rightarrow \ \mathbf{x}' \ *^\# \ \text{fact}(\mathbf{p}(\mathbf{x}'))$
- (5) $\text{FACT}(\mathbf{x}') \ \rightarrow \ \text{FACT}(\mathbf{p}(\mathbf{x}'))$
- (6) $\text{FACT}(\mathbf{x}') \ \rightarrow \ \mathbf{P}(\mathbf{x}')$

The unsorted version of cycle $\{(5)\}$ is not compatible with any well-founded and stable ordering \sqsupset on (unsorted) terms.

Throwing away all sort information is not the only ‘unsorted way’. We could also work with unsorted versions of \mathcal{C} and $\mathcal{A}_{\mathcal{C}}(\mathcal{R})$ coming from more accurate transformations Γ like the one by Ölveczky and Lysne (40).

7. Narrowing order-sorted dependency pairs

The technique of OS-DPs is a powerful way to prove termination of OS-TRSs. However, as in the standard case (5), there can be examples where the automation of the method fails due to the over-estimation of the arcs that connect two OS-DPs in the estimated OS-DG. The problem is that with the EOS-DG, we may connect more dependency pairs than needed. The over-estimation comes when an OS-DP $u_i \rightarrow v_i$ is followed in a chain by a second one $u_{i+1} \rightarrow v_{i+1}$ and v_i and u_{i+1} are not directly unifiable, i.e., at least one rule of \mathcal{R} is needed to reduce $\sigma(v_i)$ to $\sigma(u_{i+1})$. Note that if the reduction from $\sigma(v_i)$ to $\sigma(u_{i+1})$ requires at least one step, i.e., $\sigma(v_i) \rightarrow_{\mathcal{R}} \sigma(v'_i) \rightarrow_{\mathcal{R}}^* \sigma(u_{i+1})$, where v'_i is a narrowing of v_i , then instead of requiring that $u_i \sqsupset v_i$, we could require $u_i \sqsupset v'_i$ if this is easier to prove.

According to the discussion above, given (possibly renamed) pairs $u \rightarrow v$ and $u' \rightarrow v'$ such that $\sigma(v) \rightarrow_{\mathcal{R}}^* \sigma(u')$ for some substitution σ , we may perform all possible (order-sorted) one-step narrowings on v in order to develop the possible reductions from $\sigma(v)$ to $\sigma(u')$. Then, we obtain new terms v_1, \dots, v_n , which are narrowings of v with unifiers θ_i for $i \in \{1, \dots, n\}$, respectively. These unifiers are also applied to the left-hand side u of the dependency pair $u \rightarrow v$. Therefore, we can replace an OS-DP $u \rightarrow v$ by all its (one-step) *narrowed pairs* $\theta_1(u) \rightarrow v_1, \dots, \theta_n(u) \rightarrow v_n$, which are called the OS-narrowings of $u \rightarrow v$.

⁵ RPO: Recursive Path Ordering, see (8).

As in the classical approach (5), an OS-DP $u \rightarrow v \in \mathcal{P}$ may only be replaced by its narrowings if the right-hand side v does not unify with any left-hand side u' of a (possibly renamed) OS-DP $u' \rightarrow v' \in \mathcal{P}$. Moreover, the term v must be *linear*. The following result shows that, under those conditions, the set of OS-dependency pairs can be replaced by their narrowings without losing correctness or completeness.

THEOREM 6 (Narrowing transformation). *Let \mathcal{R} be an OS-TRS and $\mathcal{P} \subseteq \mathcal{P}^\sharp(\Sigma, \mathcal{X})$ be a finite set of pairs such that $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$ is sort-decreasing. Let $u \rightarrow v \in \mathcal{P}$ be such that v is linear and for all $u' \rightarrow v' \in \mathcal{P}$ (with possibly renamed variables) the terms v and u' are not unifiable. Let*

$$\mathcal{P}' = (\mathcal{P} - \{u \rightarrow v\}) \cup \{u' \rightarrow v' \mid u' \rightarrow v' \text{ is an OS-narrowing of } u \rightarrow v\}$$

There is an infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain iff there is an infinite minimal $(\mathcal{R}, \mathcal{P}')$ -chain.

PROOF. The proof of this theorem is analogous to the proof of Theorem 31 in (20), which we adapt here. For the *only if* part, we prove that given a minimal $(\mathcal{R}, \mathcal{P})$ -chain “ $\dots, u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \dots$ ”, there is a narrowing v' of v with the order-sorted mgu⁶ μ such that “ $\dots, u_1 \rightarrow v_1, \mu(u) \rightarrow v', u_2 \rightarrow v_2, \dots$ ” is also a minimal $(\mathcal{R}, \mathcal{P}')$ -chain. Hence, every infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain yields a minimal $(\mathcal{R}, \mathcal{P}')$ -chain.

If “ $\dots, u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \dots$ ” is a minimal chain, then there is an OS-substitution σ such that

1. every instantiated right-hand side reduces to the instantiated left-hand side of the next pair in the chain, and
2. all instantiated right-hand sides are \mathcal{R} -terminating (note that sort-decreasingness of $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$ together with Propositions 3 and 6 ensure that we do not need to further require \mathcal{R} -termination of all proper subterms of \mathcal{R}).

Assume that σ is such that the length of the sequence $\sigma(v) \rightarrow_{\mathcal{R}}^* \sigma(u_2)$ is *minimal*. Since v and u_2 do not unify, we have $\sigma(v) \neq \sigma(u_2)$. Hence, there is a term q such that $\sigma(v) \rightarrow_{\mathcal{R}} q \rightarrow_{\mathcal{R}}^* \sigma(u_2)$. We consider two cases:

1. The reduction $\sigma(v) \rightarrow_{\mathcal{R}} q$ takes place within a binding of σ , i.e., there is a term r , a variable position $p \in \mathcal{Pos}(v)$, and a variable x of sort s such that $v|_p = x$, $q = v[r]_p$ and $\sigma(x) \rightarrow_{\mathcal{R}} r$. Since v is linear, x occurs only once in v . Thus, $q = \sigma'(v)$ for the substitution σ' with $\sigma'(x) = r$ and $\sigma'(y) = \sigma(y)$ for all variables $y \neq x$. Here, sort-decreasingness of $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$ is essential to guarantee that σ' is well-defined in that way: since $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$ is sort-decreasing, we have that $LS(\sigma(x)) \geq LS(r)$. Note that $\sigma(x)$ is a term of sort s (where s is the sort of x), i.e., $\sigma(x) \in \mathcal{T}(\Sigma, \mathcal{X})_s$. Hence, $r \in \mathcal{T}(\Sigma, \mathcal{X})_s$ as well and binding r to x in the new OS-substitution σ' is correct. Now, the proof proceeds as in (20): since all occurrences of pairs in the sequence are variable disjoint, σ' behaves as σ for all pairs except $u \rightarrow v$. We have:

$$\sigma'(v_1) = \sigma(v_1) \rightarrow_{\mathcal{R}}^* \sigma(u) \rightarrow_{\mathcal{R}}^* \sigma'(u)$$

and

$$\sigma'(v) = q \rightarrow_{\mathcal{R}}^* \sigma(u_2) = \sigma'(u_2).$$

Note that, by minimality, $\sigma(v)$ is \mathcal{R} -terminating and, since $\sigma(v) \rightarrow q$, the term q is \mathcal{R} -terminating as well. Therefore,

⁶In the order-sorted case, the notion of ‘the mgu’ of two terms u and v is replaced by that of a *complete set of unifiers* \mathcal{U} , in such a way that given a unifier θ of u and v , there is $\mu \in \mathcal{U}$ and a substitution τ such that $\theta(x) = \tau(\mu(x))$ for all variables x . Since this is not relevant for our proof, our discourse follows the usual (unsorted) style.

$\sigma'(x) = q$ is \mathcal{R} -terminating and σ' satisfies the two conditions above. Since the length of the sequence $\sigma'(v) \rightarrow_{\mathcal{R}}^* \sigma'(u_2)$ is shorter than the sequence $\sigma(v) \rightarrow_{\mathcal{R}}^* \sigma'(u_2)$, we obtain a contradiction.

2. The reduction $\sigma(v) \rightarrow_{\mathcal{R}} q$ ‘touches’ v , i.e., there is a nonvariable position $p \in \mathcal{Pos}(v)$, and a rewrite rule $l \rightarrow r$ such that $\sigma(v|_p) = \theta(l)$, for some OS-substitution θ and

$$\sigma(v) = \sigma(v)[\sigma(v|_p)]_p = \sigma(v)[\theta(l)]_p \rightarrow_{\mathcal{R}} \sigma(v)[\theta(r)]_p = q$$

Since we can assume that variables in l are fresh, we can extend σ to behave like θ on variables in l . Thus, $\sigma(l) = \sigma(v|_p)$, i.e., l and $v|_p$ unify and there is an OS-mgu μ and an OS-substitution τ satisfying $\sigma(x) = \tau(\mu(x))$ for all variables x . We have that v narrows to $\mu(v)[\mu(r)]_p = v'$ with OS-unifier μ . Again, we can extend σ to behave like τ on the variables of $\mu(u)$ and v' . Therefore,

$$\sigma(v_1) \rightarrow^* \sigma(u) = \tau(\mu(u)) = \sigma(\mu(u))$$

and

$$\begin{aligned} \sigma(v') &= \tau(v') \\ &= \tau(\mu(v))[\tau(\mu(r))]_p \\ &= \sigma(v)[\sigma(r)]_p \\ &= \sigma(v)[\theta(r)]_p \\ &= q \\ &\rightarrow_{\mathcal{R}}^* \sigma(u_2) \end{aligned}$$

Hence, “ $\dots, u_1 \rightarrow v_1, \mu(u) \rightarrow v', u_2 \rightarrow v_2, \dots$ ” is also a minimal chain.

The *if* part is completely analogous to the ‘completeness’ part of (20, Theorem 31). \square

After narrowing the OS-DPs in $\text{OSDP}(\mathcal{R})$ we obtain a new set of narrowed pairs from which we can build a *narrowed OS-DG* and check termination as usual.

EXAMPLE 13. *For cycle $\{\text{FACT}(x') \rightarrow \text{FACT}(p(x'))\}$ in Example 12, since $\text{FACT}(x')$ and $\text{FACT}(p(x'))$ (where x' is a variable of sort NzNat) do not have any order-sorted unifier, we can apply Theorem 6 to $\text{FACT}(x') \rightarrow \text{FACT}(p(x'))$ to obtain the narrowed pair $\text{FACT}(s(x)) \rightarrow \text{FACT}(x)$, where x is a variable of sort Nat . Thanks to this narrowing transformation, we can now prove termination of FACTORIAL by using the subterm criterion discussed in Section 8 below.*

The following example shows that, without sort-decreasingness, Theorem 6 does not hold.

EXAMPLE 14. *Consider the following terminating and non-sort-decreasing OS-TRS:*

```
fmod EXAMPLE_14 is
  sorts S1 S2 S .
  subsorts S1 < S .
  subsorts S2 < S .
  op a : -> S1 .
  op b : -> S2 .
  op f : S1 -> S1 .
  op g : S -> S .
  op h : S -> S .
  var X : S1 .
  eq g(b) = f(a) .
  eq a = b .
endfm
```

and the set of pairs $\mathcal{P} = \{\text{H}(f(X)) \rightarrow \text{H}(g(X))\}$. Note that there is an infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain if we consider the OS-

substitution $\sigma(\mathbf{x}) = \mathbf{a}$:

$$\underline{H(\mathbf{f}(\mathbf{a}))} \rightarrow_{\mathcal{P}} \underline{H(\mathbf{g}(\mathbf{a}))} \rightarrow_{\mathcal{R}} \underline{H(\mathbf{g}(\mathbf{b}))} \rightarrow_{\mathcal{R}} \underline{H(\mathbf{f}(\mathbf{a}))} \rightarrow_{\mathcal{P}} \dots$$

However, since $\underline{H(\mathbf{g}(\mathbf{x}))}$ is not OS-narrowable, we would obtain $\mathcal{P}' = \emptyset$ in Theorem 6, thus making the previous infinite chain unfeasible as an $(\mathcal{R}, \mathcal{P}')$ -chain.

8. Subterm criterion

In (25), Hirokawa and Middeldorp introduced a *subterm criterion* which permits ignoring certain cycles of the dependency graph. A *simple projection* for a set of (unsorted) marked pairs $\mathcal{P} \subseteq \mathcal{P}^\sharp(\mathcal{F}, \mathcal{X})$ is a mapping π that assigns to every k -ary marked symbol f^\sharp in \mathcal{P} an argument position $i \in \{1, \dots, k\}$. The mapping that assigns to every term $f^\sharp(t_1, \dots, t_k) \in \mathcal{T}^\sharp(\mathcal{F}, \mathcal{X})$ its argument position $\pi(f^\sharp)$ is also denoted by π . In the following result, given a simple projection π , and $\mathcal{P} \subseteq \text{OSDP}(\mathcal{R})$, we let $\pi(\mathcal{P}) = \{\pi(u) \rightarrow \pi(v) \mid u \rightarrow v \in \mathcal{P}\}$. Note that $u, v \in \mathcal{T}^\sharp(\Sigma, \mathcal{X})$, but $\pi(u), \pi(v) \in \mathcal{T}(\Sigma, \mathcal{X})$.

THEOREM 7 (Subterm criterion). *Let \mathcal{R} be an OS-TRS and $\mathcal{P} \subseteq \mathcal{P}^\sharp(\Sigma, \mathcal{X})$ be a finite set of pairs such that $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$ is sort-decreasing. Let π be a simple projection such that $\pi(\mathcal{P}) \subseteq \triangleright$, and $\pi(\mathcal{P}) \cap \triangleright \neq \emptyset$. Let $\mathcal{P}_{\pi, \triangleright} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \triangleright \pi(v)\}$. There is an infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain if and only if there is an infinite minimal $(\mathcal{R}, \mathcal{P} - \mathcal{P}_{\pi, \triangleright})$ -chain.*

PROOF. (Sketch) The *if* part is obvious. For the *only if* part, we proceed by contradiction. Assume that there is an infinite $(\mathcal{R}, \mathcal{P})$ -chain A , but there is no infinite $(\mathcal{R}, \mathcal{P} - \mathcal{P}_{\pi, \triangleright})$ -chain. Due to the finiteness of \mathcal{P} , we can assume that there is $\mathcal{P}' \subseteq \mathcal{P}$ such that A has a tail B

$$t_1 \rightarrow_{\mathcal{R}}^* t'_1 \rightarrow_{\mathcal{P}'} t_2 \rightarrow_{\mathcal{R}}^* t'_2 \rightarrow_{\mathcal{P}'} \dots,$$

where all pairs in \mathcal{P}' are used infinitely often in B . Furthermore, since A is minimal, t_1 is \mathcal{R} -terminating. No pair $u \rightarrow v \in \mathcal{P}'$ satisfies that $u \triangleright v$. Otherwise, we could proceed as in the proof of (27, Theorem 8) to obtain a contradiction with the well-foundedness of \triangleright (note that the sequence B above would correspond to an infinite \mathcal{P}' -minimal sequence in the sense of (27), i.e., an infinite $(\mathcal{R}, \mathcal{P})$ -chain whose initial term is \mathcal{R} -terminating and where all pairs in \mathcal{P} are applied infinitely often). According to Proposition 1, the commutation property $\triangleright \circ \rightarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}} \circ \triangleright$, which is essential in Hirokawa and Middeldorp's proof, also holds in the order-sorted case for sort-decreasing OS-TRSs (and by Proposition 6, we only need to apply rules from $\mathcal{A}_{\mathcal{P}}(\mathcal{R})$, which is sort-decreasing).

Hence, as in the proof of Theorem 5, we conclude that $\mathcal{P}' \subseteq \mathcal{P}_{\pi, \triangleright}$. Hence, B is an infinite minimal $(\mathcal{R}, \mathcal{P} - \mathcal{P}_{\pi, \triangleright})$ -chain. This leads to a contradiction. \square

An interesting aspect of the subterm criterion formalized by Theorem 7 is that (in contrast to Theorem 5), we do *not* care about the rewrite rules in the TRS. This often leads to simpler proofs.

EXAMPLE 15. *Consider again the OS-TRS FACTORIAL in Figure 1 and the OS-DPs in Example 12. The SCC $\mathcal{C} = \{(1)\}$ can be discarded by using Theorem 7 together with the simple projection $\pi_1(+^\sharp) = 1$: note that $\mathcal{C} - \mathcal{C}_{\pi_1, \triangleright} = \emptyset$. The SCC $\mathcal{C}' = \{(3)\}$ can be handled in a similar way with the simple projection $\pi_2(*^\sharp) = 1$. After applying the narrowing transformation to the SCC $\{(5)\}$, we obtained a new SCC $\mathcal{C}'' = \{\text{FACT}(\mathbf{s}(\mathbf{x})) \rightarrow \text{FACT}(\mathbf{x})\}$ (see Example 13). Again, it can be discarded by using Theorem 7 together with the simple projection $\pi_3(\text{FACT}) = 1$. Thus, termination of FACTORIAL is proved.*

The following example shows that, without sort-decreasingness, Theorem 7 does not hold.

EXAMPLE 16. *Consider again the (terminating) OS-TRS \mathcal{R} in Example 7 and the set of pairs $\mathcal{P} = \{H(c(\mathbf{a})) \rightarrow H(\mathbf{a})\}$ where $H : S \rightarrow S$. Since $\pi(H(c(\mathbf{a}))) = c(\mathbf{a}) \triangleright \mathbf{a} = \pi(H(\mathbf{a}))$ for the simple projection π which takes the only argument of H , we could claim that there is no infinite minimal $(\mathcal{R}, \mathcal{P})$ -chain. Unfortunately, this is not true:*

$$\underline{H(c(\mathbf{a}))} \rightarrow_{\mathcal{P}} \underline{H(\mathbf{a})} \rightarrow_{\mathcal{R}} \underline{H(c(\mathbf{a}))} \rightarrow_{\mathcal{P}} \dots$$

9. Implementation and benchmarks

The techniques described in this paper have been implemented as part of the termination tool MU-TERM, which is written in the functional language HASKELL. As far as we know, this is the first time that intrinsic methods for proving termination of OS-TRSs are implemented as part of a termination tool.

In order to evaluate our techniques, we have considered OS-TRSs taken from the existing literature about termination of OS-TRSs: basically (21; 35; 40; 43) and this paper. This amounts to considering 22 examples; most of them are not terminating (viewed as TRSs). Three of them are not sort-decreasing; thus, we cannot handle them. We solve 17 of the remaining 19 in 0.1 s (global time). The two examples which cannot be currently handled correspond to a nonterminating MS-TRS (proposed by Zantema for discussing confluence of MS-TRSs, actually, see (43, Section 3.6)) and an MS-version of Toyama's example which was proved terminating by Zantema by means of an 'ad-hoc' proof see (43, Section 3.3).

In particular, we solve all the (thirteen) examples proposed (and handled by hand) in the two papers which specifically address termination of OS-TRSs, i.e., (21; 40). Actually, all of them were managed very easily by MU-TERM by either proving that there is no OS-DP, or that there is no cycle in the OS-DG, or by using the subterm criterion with the existing cycles. No reduction pair was computed actually. More details can be found here:

<http://zenon.dsic.upv.es/muterm/benchmarks/benchmarks-ostrs.html>

10. Related work and conclusions

Termination of an OS-TRS \mathcal{R} is obviously guaranteed if the *underlying* TRS $\Theta(\mathcal{R})$ is terminating. For many years, this was the only way to deal with termination of OS-TRSs. The first work envisaging nontrivial methods for proving termination of order-sorted rewriting is (21), where Gnaedig proposes an extension of the lexicographic path ordering which can prove termination of OS-TRSs when $\Theta(\mathcal{R})$ is nonterminating. Ölveczky and Lysne introduced a transformation from OS-TRSs into ordinary TRSs which can be used to prove termination of OS-TRSs; their transformation strictly subsumes Gnaedig's technique (40).

The role of sorts in termination of rewriting has also been considered in the *many-sorted* setting. Although MS-TRSs are a particular case of OS-TRSs, some interesting results have been formulated for MS-TRSs only. Termination of MS-TRSs was investigated by Zantema (43) as an auxiliary technique for proving termination of TRSs (although he mentioned no systematic method for proving termination of MS-TRSs; however, see (13)). Zantema defined the *persistency* of a computational property of MS-TRSs as a property which is *not* affected by the removal of sort information. This means that termination of persistent MS-TRSs \mathcal{R} can be characterized as termination of the underlying TRS $\Theta(\mathcal{R})$. Both Zantema and Aoto gave conditions for persistency of termination for restricted classes of MS-TRSs (3; 43). In Section 4 we have proven that OS-DPs characterize termination of sort-decreasing OS-TRSs. Since MS-TRSs are sort-decreasing OS-TRSs, our results improve on the aforementioned works.

Our main motivation has been the serious need for intrinsic methods to automatically prove termination of rewriting-based programs supporting types and subtypes, which is currently quite hard. Our main contributions to address this need can be summarized as follows. We have given a correct and complete method for proving termination of MS- and OS-TRSs based on a new generalization of the DP method to OS-TRSs (including the subterm criterion and the narrowing transformation). This is a nontrivial generalization, since the subtleties derived from the mechanism of order-sorted rewriting have to be carefully taken into account at many levels (definition of OS-DP, completeness results, estimation of the dependency graph, etc.). Furthermore, the definition of OS-DP also applies to TRSSs (viewed as one-sorted MS-TRSs), leading to a more refined version of DP thanks to the use of narrowing in the definition of DP.

EXAMPLE 17. Consider the following TRS \mathcal{R} :

$$\begin{aligned} g(f(b, Y), X, Y) &\rightarrow g(f(a, X), X, X) \\ f(c, X) &\rightarrow X \\ f(c, X) &\rightarrow f(f(a, c), c) \\ f(c, X) &\rightarrow g(f(a, X), X, X) \\ g(a, X, Y) &\rightarrow f(c, X) \end{aligned}$$

This TRS has the following ‘standard’ dependency pairs:

- (1) $G(f(b, Y), X, Y) \rightarrow G(f(a, X), X, X)$
- (2) $G(f(b, Y), X, Y) \rightarrow F(a, X)$
- (3) $F(c, X) \rightarrow F(f(a, c), c)$
- (4) $F(c, X) \rightarrow F(a, c)$
- (5) $F(c, X) \rightarrow G(f(a, X), X, X)$
- (6) $F(c, X) \rightarrow F(a, X)$
- (7) $G(a, X, Y) \rightarrow F(c, X)$

and a single SCC $\mathcal{C} = \{(1), (3), (5), (7)\}$. A powerful termination tool like TTT (27), which implements the dependency pair method as described in (25; 26; 27), fails to achieve a proof; more concretely, in few seconds it explicitly says that TTT cannot find a proof⁷. In particular, the application of the standard narrowing transformation in (5; 20) to the DPs in \mathcal{C} is of no help in this case.

In contrast, by applying the new definition of OS-DP (Definition 4) to the TRS \mathcal{R} viewed as a single-sorted MS-TRS, we obtain a single dependency pair:

$$G(a, X, Y) \rightarrow F(c, X)$$

but the dependency graph contains no cycle. The proof of termination of \mathcal{R} is immediate now (it can be automatically obtained with MU-TERM).

We have also proposed the notion of applicable rule, which is specific to the OS-DP approach, even though it has some connections with that of usable rule in the unsorted case. And we have illustrated our method on nontrivial examples, including FACTORIAL, an example for which we are not aware of any other existing method that can prove its termination. Our first set of benchmarks on OS-TRS termination problems from the literature show that our method is quite powerful already, even for the few adaptations of DP features that we provide in this paper.

Much work remains ahead. First of all, the OS-DP method presented here should be further generalized to deal with: (i) rewriting modulo axioms; and (ii) context-sensitive rewriting. The natural starting point for these generalizations is the already existing body of literature on (i)-(ii) for the DP method (1; 2; 16; 36). Second, orderings for dealing with order-sorted dependency pairs should be

⁷ APPROVE (18) obtains a proof which is based on the theory developed in (19), which is beyond the scope of this paper. Still, we believe that our new definition would also be useful in this more powerful setting.

investigated in detail. Third, the OS-DP method should be applied not only to the termination of subtype polymorphic functions as done here, but also to that of *parametrically polymorphic functions*. The key points to observe are that: (i) parameterized order-sorted theories (37) provide a very rich version of parametric polymorphism in which the parameters are not just types, but entire theories (this is supported by languages such as OBJ, Cafe-OBJ and Maude); and (ii) the *pushout semantics* by which parameterized theories are instantiated, should allow us to decompose the termination of an instantiated parameterized module into that of the rules in the uninstantiated module and those in the instance of its parameter, provided adequate modularity results in the spirit of those surveyed in, e.g., (39) are available for the union of theories computed by the pushout. Fourth, we plan to use MU-TERM with this enhanced capability for proving termination of OS-TRSs as a backend tool for MTT (11). In particular, the latest alpha version of MTT already supports a new transformation from membership equational logic theories to OS-TRSs (35) which will make the method of OS-DPs applicable to the wider and more expressive class of membership equational programs, which support types, subtypes, conditional rules, and partiality. Finally, these tool implementations will make possible extensive experimentation with our OS-DP method to both perfect it and assess its practical applications.

Acknowledgments

We thank the referees for many useful remarks.

References

- [1] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In N. Garg and S. Arun-Kumar, editors *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, LNCS 4337:297-308, Springer-Verlag, Berlin, 2006.
- [2] B. Alarcón, R. Gutiérrez, and S. Lucas. Improving the Context-Sensitive Dependency Graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.
- [3] T. Aoto. Solution to the Problem of Zantema on a Persistent Property of Term Rewriting Systems. *Journal of Functional and Logic Programming*, 2001(11):1-20, 2001.
- [4] T. Aoto and T. Yamada. Dependency Pairs for Simply Typed Term Rewriting. In J. Giesl, editor, *Proc. of the 16th International Conference on Rewriting Techniques and Applications, RTA'05*, LNCS 3467:120-134, Springer-Verlag, Berlin, 2005.
- [5] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.
- [6] P. Borovanský, C. Kirchner, H. Kirchner, and P.-E. Moreau. ELAN from a rewriting logic point of view. *Theoretical Computer Science*, 285:155–185, 2002.
- [7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All About Maude – A High-Performance Logical Framework. *Lecture Notes in Computer Science* 4350, 2007.
- [8] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.
- [9] N. Dershowitz. Termination by Abstraction. In B. Demoen and V. Lifschitz, editors, *Proc. of 20th International Conference on Logic Programming, ICLP'04*, LNCS 3132:1-18, Springer-Verlag, Berlin, 2004.
- [10] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Termination of Membership Equational Programs. In P. Seftoft and N. Heintze, editors, *Proc. of ACM SIGPLAN 2004 Symposium PEPM'04*, pages 147–158. ACM Press, 2004.
- [11] F. Durán, S. Lucas, and J. Meseguer. MTT: The Maude Termination Tool. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proc. of the Fourth International Joint Conference on Automated Reasoning*,

- IJCAR'08*, LNAI to appear, Springer-Verlag, Berlin, 2008. Available at <http://www.lcc.uma.es/~duran/MTT>.
- [12] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, 21(1-2):59–88, 2008.
- [13] J. Endrullis, J. Waldmann, and H. Zantema. Matrix Interpretations for Proving Termination of Term Rewriting. *Journal of Automated Reasoning* 40(2-3):195–220, 2008.
- [14] K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.
- [15] J. Giesl, T. Arts, and E. Ohlebusch. Modular Termination Proofs for Rewriting Using Dependency Pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.
- [16] J. Giesl and D. Kapur. Dependency Pairs for Equational Rewriting. In A. Middeldorp, editor, *Proc of the 12th International Conference on Rewriting Techniques and Applications, RTA'01*, LNCS 2051:93–107, Springer Verlag, Berlin, 2001.
- [17] J. Giesl, S. Swiderski, P. Schneider-Kamp, and R. Thiemann. Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages. In F. Pfenning, editor, *Proc of the 18th International Conference on Rewriting Techniques and Applications, RTA'06*, LNCS 4098:297–312, Springer Verlag, Berlin, 2006.
- [18] J. Giesl, P. Schneider-Kamp, and R. Thiemann. APROVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281–286, Springer-Verlag, Berlin, 2006. Available at <http://www-i2.informatik.rwth-aachen.de/APROVE>.
- [19] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In F. Baader and A. Voronkov, editors, *Proc. of XI International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04*, volume 3452 of *Lecture Notes in Artificial Intelligence*, pages 301–331, Montevideo, Uruguay, 2004. Springer-Verlag.
- [20] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning*, 37(3):155–203, 2006.
- [21] I. Gnaedig. Termination of Order-sorted Rewriting. In H. Kirchner and G. Levi, editors, *Proc. of the 3th International Conference on Algebraic and Logic Programming, ALP'92*, LNCS 632:37–52, Springer-Verlag, Berlin, 1992.
- [22] J. Goguen, J.-P. Jouannaud, and J. Meseguer. Operational Semantics for Order-Sorted Algebra. In *Proc. of Automata, Languages and Programming, 12th Colloquium, ICALP'85*, LNCS 194:221–23, Springer-Verlag, Berlin, 1985.
- [23] J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.
- [24] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.
- [25] N. Hirokawa and A. Middeldorp. Dependency Pairs Revisited. In V. van Oostrom, editor, *Proc. of 15th International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:249–268, Springer-Verlag, 2004.
- [26] N. Hirokawa and A. Middeldorp. Automating the Dependency Pair Method. *Information and Computation*, 199:172–199, 2005.
- [27] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474–511, 2007.
- [28] P. Hudak, S. Peyton-Jones, and P. Wadler. Report on the Functional Programming Language Haskell: a non-strict, purely functional language. *SIGPLAN Notices*, 27:1–164, 1992.
- [29] S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering. University of Illinois at Urbana-Champaign. Unpublished manuscript, 1980.
- [30] C. Kirchner, H. Kirchner, and J. Meseguer. Operational Semantics of OBJ3. In T. Lepistö and A. Salomaa, editors, *Proc. of 15th Intl. Coll. on Automata, Languages and Programming, ICALP'88*, LNCS 317:287–301, Springer-Verlag, Berlin, 1988.
- [31] K. Kusakari, M. Nakamura, and Y. Toyama. Argument Filtering Transformation. In G. Nadathur, editor, *Proc. of International Conference on Principles and Practice of Declarative Programming, PPDP'99*, LNCS 1702:47–61, Springer-Verlag, 1999.
- [32] K. Kusakari and M. Sakai. Enhancing dependency pair method using strong computability in simply-typed term rewriting. *Applicable Algebra in Engineering, Communication and Computing* 18:407–431, 2007.
- [33] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of 15th International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200–209, Springer-Verlag, Berlin, 2004. Available at <http://zenon.dsic.upv.es/muterm>.
- [34] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In *Proc. of 3rd International Conference on Principles and Practice of Declarative Programming, PPDP'01*, pages 82–93, ACM Press, 2001.
- [35] S. Lucas and J. Meseguer. Operational Termination of Membership Equational Programs: the Order-Sorted Way. In G. Rosu, editor, *Proc. of the 7th International Workshop on Rewriting Logic and its Applications, WRLA'08*, Electronic Notes in Theoretical Computer Science, to appear, 2008.
- [36] C. Marché and X. Urbain. Modular and incremental proofs of AC-termination. *Journal of Symbolic Computation*, 38(1):873–897, 2004.
- [37] J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *Proc. of WADT'97*, volume 1376 of *Lecture Notes in Computer Science*, LNCS 1376:18–61, Springer-Verlag, 1998.
- [38] U. Nilsson and J. Maluszynski. *Logic, Programming and Prolog* (2nd ed.) John Wiley & Sons, 1995.
- [39] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.
- [40] P.C. Ölveczky and O. Lysne. Order-Sorted Termination: The Unsorted Way. In M. Hanus and M. Rodríguez-Artalejo, editors, *Proc. of the 5th International Conference on Algebraic and Logic Programming, ALP'96*, LNCS 1139:92–106, Springer-Verlag, Berlin, 1996.
- [41] P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann. Automated Termination Analysis for Logic Programs by Term Rewriting. In R. K. Shyamasundar, editor, *Proc. of the 16th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR '06*, LNCS 4407:177–193, Springer-Verlag, Berlin, 2007.
- [42] TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003.
- [43] H. Zantema. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.